**Лекція 2**

**2Введення до UML. Діаграма класів**

UML (Unified Modeling Language) – уніфікована мова моделювання – це система позначень, що застосовується для об'єктно-орієнтованого аналізу та проектування, це мова діаграм або позначень для специфікації, візуалізації та документації моделі об'єктно-орієнтованих програмних систем. UML не є методом розробки, тобто він не визначає послідовність дій при розробці ПЗ. Він допомагає описати свою ідею і взаємодіяти з іншими розробниками системи. UML управляється Object Management Group (OMG) і є промисловим стандартом, що описує моделі ПЗ. Повний опис UML є за адресою:

https://www.omg.org/spec/UML/2.5.1/PDF

UML визначає різні види діаграм, наприклад, такі:

1. Діаграма прецедентів.
2. Діаграма класів.
3. Діаграма об'єктів.
4. Діаграма послідовностей.
5. Діаграма взаємодії.
6. Діаграма станів.
7. Діаграма активності.
8. Діаграма розгортання.

*Діаграма класів*

Це діаграма, на якій показані класи, інтерфейси та відносини між ними. Головний елемент діаграми класів – клас. При проектуванні об'єктноорієнтованих систем діаграми класів обов'язкові.

Класи використовуються в процесі аналізу предметної області для складання словника предметної області системи, що розробляється. Це можуть бути як абстрактні поняття предметної області, так і класи, на які спирається розробка та які описують програмні або апаратні сутності.

Діаграма класів є набором статичних, декларативних елементів моделі.

Класи зображуються у вигляді прямокутників, зазвичай розділених на дві або три частини (рис. 4.1). У верхній частині знаходиться ім'я класу. Середня частина містить список змінних класу, а нижня частина – методи класу.

Символи, зазначені перед кожною змінною або методом, представляють собою індикатори видимості (visibility indicators). Можливі індикатори видимості та їх значення представлені в таблиці 1. Змінній може присвоюватися відповідне її типу значення за допомогою знака «=» та деякої величини (рис.

4.1).



Рис. 4.1. Приклад оформлення класу на діаграмі класів UML

Таблиця 4.1 Індикатори видимості

|  |  |
| --- | --- |
| Індикатор | Значення |
| + | відкритий (public)public)) |
| # | захищений (public)protec)ted)) |
| – | закритий (public)private) |
| ~ | для модуля (public)pac)kage) |

На схемах UML слово, укладене в кутові лапки (дужки), називається *стереотипом*. Стереотип описує те, що за ним випливає. Наприклад, стереотип constructor вказує на те, що наступний за ним метод(и) являє собою конструктор класу. Стереотип misc вказує, що наступний за ним метод(и) – регулярний.

У класі можливий спеціальний елемент – (...) – еліпсис (відображається не у всіх пакетах ПЗ). Він вказує на те, що в класі є додаткові змінні або методи, які не показані на діаграмі.

Як правило, немає необхідності (або просто незручно) показувати клас детально. Тому зображення класу може складатися тільки з двох частин – імені класів та методів або з однієї частини – тільки імені класу (рис. 4.2).

 б) а)

Рис. 4.2. Різновиди зображення класу

Як у методів, так і у змінних можуть бути відсутні індикатори видимості. Однак це не означає, що їх немає.

Інтерфейси зображуються так само, як і класи, тільки у верхній частині імені вказується стереотип interface (рис. 4.3).



Рис. 4.3. Представлення інтерфейсу

На рис. 4.4, а, б показаний приклад абстрактного класу BaseIO, що є суперкласом для класів BaseFile та TextTerminal. Ім'я абстрактного класу, а також його абстрактні методи записуються курсивом.

 б) а) 

в)

Рис. 4.4. Успадкування підкласів від суперкласу (базового, абстрактного) (а, б) та реалізація класом інтерфейсу (в)

Суцільна лінія зі стрілкою у вигляді замкненого контуру вказує на те, що даний підклас успадковується від суперкласу (рис. 4.4, а, б). Якщо ж клас реалізує інтерфейс, то використовується пунктирна або штрих-пунктирна лінія зі стрілкою у вигляді замкнутого контуру (рис. 4.4, в) – клас GeoPoint реалізує інтерфейс GeoObject.

Якщо між класами використовується звичайна лінія або лінія зі стрілками, то такого типу відносини називаються *асоціаціями*. Разом з асоціаціями можуть зазначатися деякі дані, що представляють інформацію про суть асоціації: ім'я асоціації, стрілки навігації, ім'я ролі, індикатор множинності.

Ім'я асоціації може бути вказано приблизно в середині з'єднання та починатися з великої літери (рис. 4.5). В кінці або на початку імені асоціації може бути зображений трикутник, який вказує напрямок, в якому слід читати асоціацію.



Рис. 4.5. Приклад оформлення асоціації

На рис. 4.5 зображена асоціація Create (Створює) між класами Factory та Product.

Стрілки на кінцях асоціацій називаються *стрілками навігації*. Вони вказують напрямок переміщення по асоціації. Наприклад, стрілка на рис. 4.5 говорить про відповідальність класу Factory за створення екземплярів об'єктів класу Product.

Природа деяких інших асоціацій менш очевидна. Для розуміння суті таких асоціацій може виникнути необхідність в наданні додаткової інформації, що стосується асоціації – задається ім'я ролі, яку кожен клас відіграє в асоціації. Ім'я ролі може бути вказано на будь-якому кінці асоціації. Імена ролей завжди записують малими літерами. Це дозволяє легко відрізнити їх від імен асоціацій.

На рис. 4.6 видно, що клас CreationRequestor бере участь в асоціації в ролі ініціатора запиту, а інтерфейс FactoryIF – в ролі творця.

Індикатор множинності є іншою характеристикою асоціації, яка часто зустрічається та повідомляє про те, скільки примірників кожного класу (об’єктів) беруть участь в асоціації. Такий індикатор може вказуватися на будьякому кінці асоціації та представляти просто число (наприклад, 0 або 1) або діапазон чисел (наприклад, 0..2). Зірочка використовується замість верхньої межі діапазону та позначає необмежену кількість випадків. Індикатор множинності 1..\* вказує принаймні на один екземпляр, 0..\* – на будь-яку кількість екземплярів. Просто \* еквівалентна 0..\*. На рис. 6 все асоціації схеми являють собою співвідношення «один до багатьох».



Рис. 4.6. Приклад асоціацій з іменами ролей

Іноді може виникнути необхідність більшої структуризації, ніж в представленій схемі з простим співвідношенням «один до багатьох». Співвідношення «один до багатьох», в якому один об'єкт містить набір інших об'єктів, називається *агрегацією*. На агрегацію вказує контур у вигляді ромба, розташований на тому кінці асоціації, який стикується з класом, що містить екземпляри іншого класу (рис. 4.7).



Рис. 4.7. Приклад агрегації

На рис. 4.7 зображено клас MessageManager. З рис. 4.7 випливає, що кожен екземпляр класу MessageManager містить нуль або більше примірників класу UserMsg.

В UML можна використовувати ще одне позначення, яке вказує на більш сильний, ніж агрегація, зв'язок. Цей зв'язок називається *композитною агрегацією*. Щоб агрегація була композитної, повинні виконуватися дві умови:

1. У якийсь момент часу екземпляри, що агрегуються, повинні належатитільки одному складеному (рос.: составному) об'єкту.
2. Деякі операції повинні передаватися у спадок від складеного об'єкта дойого екземплярів, що агрегуються. Наприклад, якщо складений об'єкт клонується, то зазвичай клонуються і його агрегуємі екземпляри. Тому новий складений об’єкт володіє власними клонами вихідних агрегованих примірників.

На рис. 4.8 представлена діаграма класів, яка містить композитну агрегацію. Об'єкти класу Document можуть містити об'єкти класу Paragraph, які можуть мати в своєму складі об'єкти класу DocChar. Завдяки композитної агрегації очевидно, що об'єкти класу Paragraph не використовують спільно об'єкти класу DocChar, а об'єкти класу Document не використовують спільно об'єкти класу

|  |  |
| --- | --- |
| Рис. 4.8. Композитна агрегація | Paragraph.  |

Розглянемо більш детально приклади з агрегацією і композицією.

На діаграмі (рис. 4.9) показано, що існує клас University, з яким буде пов'язаний один екземпляр об'єкту та який інкапсулює 20-ть об'єктів типу Department. Сам клас Department такий, що реалізації відповідного класу може бути в межах University 20, однак по відношенню до об'єктів класу Professor їх може бути скільки завгодно. Клас Department агрегує масив з 5-ти покажчиків на об'єкти типу Professor. Кодом на С ++ дана ситуація буде описана наступним чином.



Рис. 4.9. Приклад агрегації та композиції

**class Professor; class Department**

**{**

 **... private:**

 **Professor\* members[5]; // Звичайна агрегація – у складі кафедр може**

 **// і не бути професорів.**

 **// Об’єкт типу Professor створюється окремо.**

 **...**

**};**

**class University**

**{**

 **... private:**

  **Department faculty[20]; // Композитна агрегація – університет**

 **// не може існувати без кафедр. // Це окрема одиниця університету, яка з ним // жорстко пов’язана.**

 **... public: void University( )**

 **{**

 **....**

 **// Composition**

 **faculty[0] = createDepartment(....); faculty[1] = createDepartment(....);**

 **....**

 **}**

**};**

Інший приклад представлений на рис. 4.10. В автомобілі (клас Car) може бути чи ні (якщо він не створений) карбюратор (об'єкт класу Carburetor). У ставку (клас Pond) може бути безліч качок. Авто не може працювати без карбюратору. Ставок може існувати й без качок.



Рис. 4.10. Приклад агрегації та композиції

Приклад коду на C++:

**// Composition**

**class Car**

**{ private:**

 **Carburetor itsCarb;**

 **...**

**};**

**// Aggregation class Pond**

**{ private:**

 **vector<Duck\*> itsDucks;**

 **...**

**};**

Приклади з життя: комп’ютер може працювати без клавіатури (агрегація), клавіатура не може працювати без клавіш (композитна агрегація).

Деякі асоціації є непрямими. Замість прямого зв'язування класів один з одним, вони зв'язуються побічно через третій клас. Наприклад, на рис. 4.11 асоціація показує, що екземпляри класу Cache посилаються на екземпляри класу Object через екземпляр класу ObjectID.

 Асоціація між класами або інтерфейсами передбачає наявність залежності, яка містить об'єктне посилання, що пов'язує два об'єкти.

Рис. 4.11. Клас асоціації Мається на увазі, що один клас або інтерфейс містить в собі посилання на інший клас або

інтерфейс, і це взаємодія відображається на діаграмі. Можливі також залежності інших видів. Для вказівки залежності більш загального вигляду, використовується пунктирна лінія. Приклад подібної залежності представлений на рис. 4.12.



Рис. 4.12. Залежність

Класи на діаграмі можуть бути організовані у пакети. Пакет зображується як великий прямокутник з маленьким прямокутником нагорі, в якому вказується ім'я пакета (рис. 4.13).



Рис. 4.13. Пакет

На рис. 4.13 показаний пакет з ім'ям ServicePackage. Перед ім'ям класу або інтерфейсу, що знаходиться всередині пакету, може вказуватися індикатор видимості. Відкриті класи доступні для класів за межами пакета, закриті – ні.

Іноді при проектуванні виявляються обставини, незрозумілі без коментаря на діаграмі. В UML коментар зображується у вигляді прямокутника з загнутим правим верхнім кутом і приєднується до того елемента діаграми, до якого вони належать.